Concurrency in Elixir

Let it crash with GenServer, Supervisor, et al.

ElixirCasually. February 21st, 2019.

@unnawut, OmiseGO

What you'll see today?

1. Umbrella and subapps
2. GenServer
3. Supervisor
4. Crash me if you can! 😱

Umbrella & sub apps

```
elixir_casually_umbrella/ # Umbrella
  apps/
    elixir_casually/ # Sub-App 1
    elixir_casually_web/ # Sub-App 2
    load_tester/ # Sub-App 3
  mix.exs
  mix.lock
```

Umbrella & sub apps

```
elixir_casually_umbrella/ # Umbrella
  apps/
    elixir_casually/ # Sub-App 1
    elixir_casually_web/ # Sub-App 2
    load_tester/ # Sub-App 3
      lib/
        load_tester.exs
      mix.exs
  mix.exs
  mix.lock
```

Umbrella & sub apps

```
elixir_casually_umbrella/ # Umbrella
  apps/
    elixir_casually/ # Sub-App 1
    elixir_casually_web/ # Sub-App 2
    load_tester/ # Sub-App 3
      lib/
        load_tester.exs
      mix.exs
  mix.exs
  mix.lock
```

Umbrella & sub apps allow us to either...

Umbrella & sub apps

```
elixir_casually_umbrella/ # Umbrella
  apps/
    elixir_casually/ # Sub-App 1
    elixir_casually_web/ # Sub-App 2
    load_tester/ # Sub-App 3
      lib/
        load_tester.exs
      mix.exs
  mix.exs
  mix.lock
```

Umbrella & sub apps allow us to either...

1. Start EACH app separately

Umbrella & sub apps

```
elixir_casually_umbrella/ # Umbrella
  apps/
    elixir_casually/ # Sub-App 1
    elixir_casually_web/ # Sub-App 2
    load_tester/ # Sub-App 3
      lib/
        load_tester.exs
      mix.exs
  mix.exs
  mix.lock
```

Umbrella & sub apps allow us to either...

1. Start EACH app separately
2. Start ALL apps as a whole

What you'll see today?

1. 👌 Umbrella and subapps
2. GenServer
3. Supervisor
4. Crash me if you can!

Task.async()

```
Task.async(fn ->
  {:ok, _} = VoterRegistry.register(citizen_id)
  {:ok, _} = Counter.increment(party_id)

  IO.puts("Voted with async: #{inspect(self())}")
  json(conn, %{success: true})
end)
```

http://12.12.12.19:5000/vote/async

GenServer

"A GenServer is a process that can be used to **keep state**, execute code **asynchronously** and so on."

https://hexdocs.pm/elixir/GenServer.html

Voter & Counter

```elixir
defmodule ElixirCasually.Voter do
  use GenServer

  def start_link(_opts), do: #...
  def register(citizen_id), do: #...
end

defmodule ElixirCasually.Counter do
  use GenServer

  def start_link(_opts), do: #...
  def increment(party_id), do: #...
end
```

VoteController

```
defmodule ElixirCasuallyWeb.VoteController do
  use Phoenix.Controller

  def create(conn, ...) do
    {:ok, _} = VoterRegistry.register(citizen_id)
    {:ok, _} = Counter.increment(party_id)

    json(conn, %{success: true})
  end
end
```

http://12.12.12.19:5000/vote/genserver

GenServer

- Handles independent workloads
- Avoids managing sync locks
- Avoids spawning new processes
- Isolates failures
- Paves the way for automatic recovery 🤔

What you'll see today?

1. 👌 Umbrella and subapps
2. 👌 GenServer
3. Supervisor
4. Crash me if you can!

Supervisor

"... provides fault-tolerance and encapsulate how our applications start and shutdown."

https://hexdocs.pm/elixir/Supervisor.html

RegistrySupervisor

```elixir
defmodule ElixirCasually.RegistrySupervisor do
  use Supervisor

  def start_link(opts), do: #...

  def init(_opts) do
    child_specs = [
      ElixirCasually.VoterRegistry, # GenServer
      ElixirCasually.Counter # GenServer
    ]

    Supervisor.init(child_specs, strategy: :one_for_one)
  end
end
```

RegistrySupervisor

Demo!

What you'll see today?

1. 👌 Umbrella and subapps
2. 👌 GenServer
3. 👌 Supervisor
4. Crash me if you can!

Crash me if you can!

http://12.12.12.19:5000

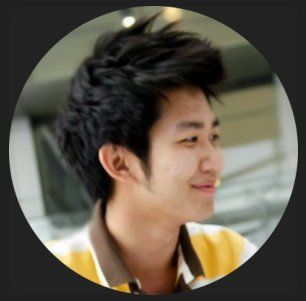
What you've seen today

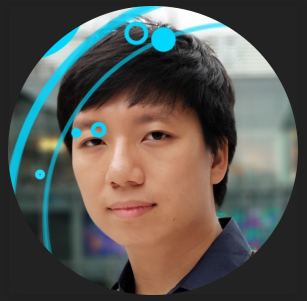
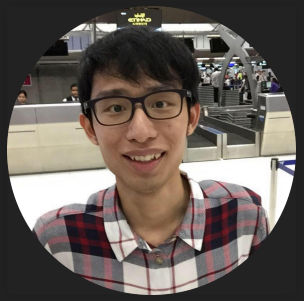
1. 👌 Umbrella and subapps
2. 👌 GenServer
3. 👌 Supervisor
4. 👌 Crash me if you can!

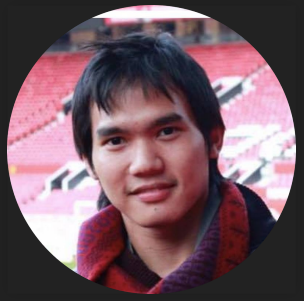
Pick Elixir for

1. Fault tolerance
2. Concurrency & distributed architecture
3. Microservices without headache

Many thanks to





References

- https://github.com/unnawut
- https://github.com/unnawut/elixir-casually-demo
- https://github.com/omisego/ewallet
- https://github.com/omisego/elixir-omg